EXHIBIT V

FILED UNDER SEAL

Contains Highly Confidential AEO and Source Code Materials

IN THE UNITED STATES DISTRICT COURT

NORTHERN DISTRICT OF CALIFORNIA

SAN FRANCISCO DIVISION

GOOGLE LLC, Plaintiff v. SONOS, INC., Defendant.	CASE NO. 3:20-cv-06754-WHA Related to CASE NO. 3:21-cv-07559-WHA

REBUTTAL EXPERT REPORT OF SAMRAT BHATTACHARJEE REGARDING NON-INFRINGEMENT OF CLAIM 13 OF U.S. PATENT NO. 9,967,615 AND OTHER ISSUES

HIGHLY CONFIDENTIAL AEO AND SOURCE CODE MATERIALS

361. Finally, Dr. Schmidt argues that NIA No. 1 would still infringe based on "the videoIDs provided by the" WatchNext servers. Specifically, each time the receiver plays a new song or video it makes a WatchNext request and receives a WatchNext response that may contain a previous, current, and next video *Id.* According to Dr. Schmidt, in Google's NIA No. 1 the WatchNext servers would continue to serve as a first set of cloud servers that provide videoIds (the alleged resource locators) to a YouTube receiver, and that are separate from the Bandaid

servers that provide the chunks of multimedia content. Schmidt Rpt., 517. Dr. Schmidt's argument is incorrect. Even assuming a videoId is a resource locator (it is not), Dr. Schmidt fails to account for the fact that Google's Non-Infringing Alternative No. 1 involves implementing Streaming Watch along with Onesie. Streaming Watch allows the Onesie agent to provide the information in a WatchNext response: "The idea here is that clients make a single request to Onesie. Onesie will then make the GetPlayer and GetWatchNext RPCs. Onesie can then stream back to the client the GetPlayer response ahead of the GetWatchNext response via chunked HTTP." GOOG-SONOSNDCA-00071671; *see also* GOOG-SONOSNDCA-00070863 ("streaming Watch is a project to merge GetPlayer and GetWatchNext into a single streaming RPC (aka GetWatch)"). Put another way, with Streaming Watch Google's NIA No. 1 would merge the functionality of the WatchNext and Bandaid servers, such that the WatchNext servers would not be a separate set of servers, as required by Claim 13.

362. Dr. Schmidt's speculation that the alternative would "not have been technically feasible" because it would create a "single point of failure" ignores the fact that Google has already implemented Onesie for YouTube applications in the non-casting use case. See e.g., GOOG-SONOSNDCA-00073431 ("Main app - Launched", "Music app - Coming soon, user-facing experiment in progress as of 2020Q1," "iOS - Launched," "HTML5 - Coming soon, implementation in progress as of 2019 Q4"); (GOOG-SONOSNDCA-00073427) (Onesie) ("Onesie was launched on the YouTube Android app in 2016/05"). And it is my understanding that Google has plans to release Onesie for the casting use case by the end of the year. *See e.g.*, GOOG-SONOSNDCA-00116349, GOOG-SONOSNDCA-00116350, GOOG-SONOSNDCA-00116353. Google also has plans to release Streaming Watch for its YouTube devices. *Id.* The fact that Google has continued to deploy and that many YouTube services have successfully been

using Onesie demonstrates that Onesie does not introduce a single point of failure, as Dr. Schmidt suggests.

363. Similarly, Dr. Schmidt's speculation that the alternative would "not have been commercially acceptable" has no factual basis. As an initial matter Onesie and Streaming Watch are architectural changes that are not "visible" to end-users. Thus, from a user perspective the experience playing YouTube and GPM would remain the same. In fact, Onesie increases the user experience by reducing latency and thereby leading to faster playback. (GOOG-SONOSNDCA-00073427) (Onesie) ("So far the project has increased fast playbacks (under 1s of latency) by 35%."). And again, the fact that Google has released Onesie and plans to release Streaming Watch further indicates that they are commercially acceptable alternatives.

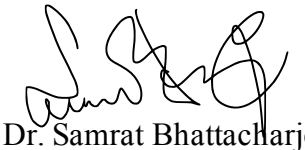
364. Lastly, Dr. Schmidt claims it is "unclear to [him] if or how 'Streaming Watch' is applicable to Google's proposal to '[s]erv[e] URLs from the same server and/or service as the content.'" Schmidt Rpt., 523. But, as I explained in my opening report, StreamingWatch is a project to merge GetPlayer and GetWatchNext into a single streaming RPC called GetWatch. Bhattacharjee Rpt., 592. In other words, Onesie alone (without using Streaming Watch) would serve URLs "from the same server and/or service as the content"). Streaming Watch is directed only at Sonos's argument that a videoId sent to the receiver by a WatchNext service is a resource locator.

365. Dr. Schmidt also points to statements in a document Streaming Watch ~~2 pager~~ (GOOG-SONOSNDCA-00071671) that say "[t]he merge of the InnerTube endpoints GetPlayer and GetWatchNext is something that has been talked about at YouTube for quite some time now" and that because the project is a "large undertaking… it is important that we agree that the results are worth the effort." Schmidt Rpt., 523. But Dr. Schmidt ignores the portions of the document

considerations. It is also my understanding that Sonos may submit an expert report corresponding to this report. I reserve the right to rebut any positions taken in that report.

394. I, Samrat Bhattacharjee, declare under penalty of perjury under the laws of the United States that the foregoing is true and correct.

DATED: July 27, 2022



Dr. Samrat Bhattacharjee, PhD.